Linear Discrimination

- Introduction
- Linear Discriminant Functions
- Two-Category Linearly Separable Case
- The Perceptron Algorithm
- Least Squares Method
- Logistic Discrimination
- Support Vector Machines

Introduction (1)

- In probabilistic methods
  - Needs to model the unknown pdfs $p(\mathbf{x}|C_i)$
    - Parametric model
      - Form of the underlying pdf is assumed to be known
      - Training samples are used to estimate density parameters $\boldsymbol{\theta}$
    - Nonparametric estimation
      - Form of the underlying pdf is NOT known
      - Training samples are used to estimate the density functions
  - No need to re-train if adding a new class
  - Stronger assumption on generative models

Introduction (2)

- Discriminant-based methods
  - Model the discriminant function $g(\mathbf{x})$ directly
    - Linear, nonlinear, or generalized linear models
  - Form of $g(\mathbf{x})$ is assumed to be known
    - Regardless of the underlying pdf describing the training data
    - The assumed form may not be optimal
      - Simpler and less computational demanding
  - Fitting needs solving optimization problem
    - e.g., initialized with random parameters, and iteratively adjusted to minimize an error function

Linear Discriminant Functions (1)

- Linear discriminant function
  - A linear combination of input features
    - $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = w_l x_l + \cdots + w_1 x_1 + w_0$
      - $\mathbf{w}$: weight vector
      - w_0: bias or threshold weight

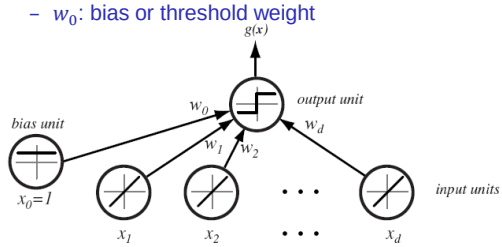


Fig. 5.1 [Duda 01]

FIGURE 5.1. A simple linear classifier having d input units, each corresponding to the values of the components of an input vector. Each input feature value x_i is multiplied by its corresponding weight w_i; the effective input at the output unit is the sum all these products, $\sum w_i x_i$. We show in each unit its effective input-output function. Thus each of the d input units is linear, emitting exactly the value of its corresponding feature value.

Linear Discriminant Functions (2)

- The decision surfaces are linear functions of $\mathbf{x}$
  - Defined by $(l-1)$-dimensional hyperplanes within the l –dimensional space

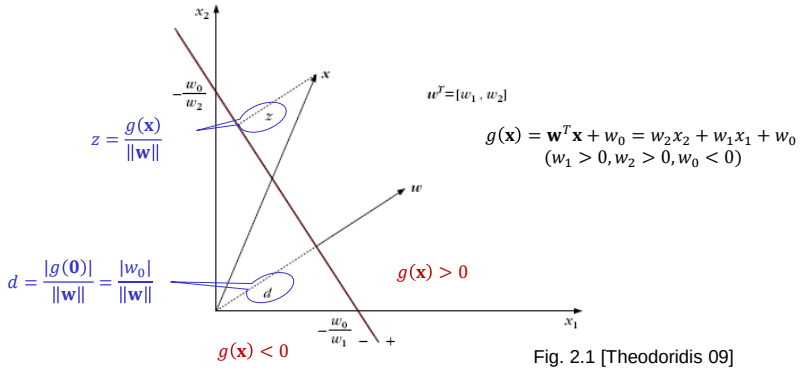


Fig. 2.1 [Theodoridis 09]

Linear Discriminant Functions (3)

- Two-category case
  - If $\mathbf{x}_1$ and $\mathbf{x}_2$ are both on the decision surface (hyperplane) H
    - $\mathbf{w}^T\mathbf{x}_1+w_0=\mathbf{w}^T\mathbf{x}_2+w_0=0 \Rightarrow \mathbf{w}^T(\mathbf{x}_1-\mathbf{x}_2)=0$
      - $\mathbf{w}$ is orthogonal to the hyperplane
  - Let
    - $\mathbf{x}=\mathbf{x}_p+z\frac{\mathbf{w}}{\|\mathbf{w}\|}$
      - where $\mathbf{x}_p$ is on H, $i.e., g(\mathbf{x}_p)=0$
  - $g(\mathbf{x})=(\mathbf{w}^T\mathbf{x}_p+w_0)+z\frac{\mathbf{w}^T\mathbf{w}}{\|\mathbf{w}\|}=z\|\mathbf{w}\|$
    - z: the signed distance from $\mathbf{x}$ to H
      - $z=\frac{g(\mathbf{x})}{\|\mathbf{w}\|}$

H divides the l-dim feature space into 2 half-spaces

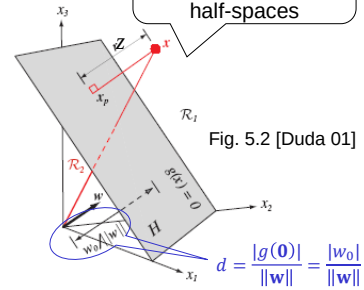


Fig. 5.2 [Duda 01]

$|g(\mathbf{x})|$ is a measure of the Euclidean distance of $\mathbf{x}$ to H

$\mathbf{w}$ determines the orientation of the decision surface
w_0 determines the location of the surface

Linear Discriminant Functions (4)

- Multicategory case (Ch2)
  - One-against-the-rest
    - Use K two-class problems
      - Each assigns $\mathbf{x}$ to C_i or not C_i
    - Class imbalance problem
      - #negative >> #positive
  - Pairwise separation
    - Use $K(K-1)/2$ classifers
      - For every pair of classes
      - Decision can be made by majority vote
    - May lead to nonlinear separation of classes

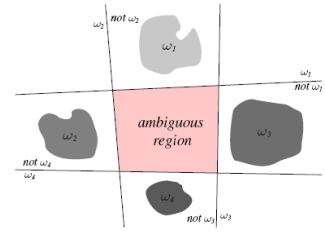
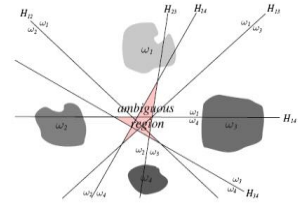


Fig. 5.3 [Duda 01]

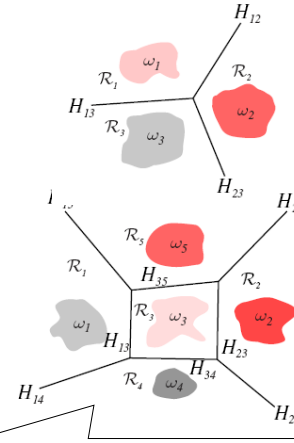


Both approaches lead to undefined regions in the feature space

Linear Discriminant Functions (5)

Fig. 5.4 [Duda 01]

- Multicategory case (cont.)
  - Linear machine
    - Define K discriminant functions
      - $g_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + w_0, i = 1, \dots, K$
      - $\mathbf{x} \rightarrow C_i$ if $g_i(\mathbf{x}) = \max_k g_k(\mathbf{x})$
    - Part of the hyperplane H_{ij} is defined by
      - $g_i(\mathbf{x}) = g_j(\mathbf{x})$
      - $(\mathbf{w}_i - \mathbf{w}_j)^T\mathbf{x} + (w_{i0} - w_{j0}) = 0$
    - If $\mathbf{x}_1$ and $\mathbf{x}_2$ are both on H_{ij}
      - $(\mathbf{w}_i - \mathbf{w}_j)^T(\mathbf{x}_1 - \mathbf{x}_2) = 0$
      - $(\mathbf{w}_i - \mathbf{w}_j)$ is normal to the hyperplane H_{ij}
    - The signed distance from $\mathbf{x}$ to H_{ij} is
      - $\frac{g_i(\mathbf{x}) - g_j(\mathbf{x})}{\|\mathbf{w}_i - \mathbf{w}_j\|}$



The decision regions for a linear machine are always singly connected and convex

Linear Discriminant Functions (6)

- Multicategory case (cont.)
  - The decision regions in a linear machine are convex
    - If $\mathbf{x}_1 \in R_i$ and $\mathbf{x}_2 \in R_i$
    - Then $\lambda\mathbf{x}_1 + (1-\lambda)\mathbf{x}_2 \in R_i$ for $0 \le \lambda \le 1$
      - $\mathbf{x}_1 \in R_i \Rightarrow \max_{j=1,\ldots,K} g_j(\mathbf{x}_1) = g_i(\mathbf{x}_1) = \mathbf{w}_i^T\mathbf{x}_1 + w_{i0}$
      - $\mathbf{x}_2 \in R_i \Rightarrow \max_{j=1,\ldots,K} g_j(\mathbf{x}_2) = g_i(\mathbf{x}_2) = \mathbf{w}_i^T\mathbf{x}_2 + w_{i0}$
      - For any j and $0 \le \lambda \le 1$
      - $g_j(\lambda\mathbf{x}_1 + (1-\lambda)\mathbf{x}_2) = \mathbf{w}_j^T(\lambda\mathbf{x}_1 + (1-\lambda)\mathbf{x}_2) + w_{j0}$
        $= \lambda(\mathbf{w}_j^T\mathbf{x}_1 + w_{j0}) + (1-\lambda)(\mathbf{w}_j^T\mathbf{x}_2 + w_{j0})$
        $\le \lambda(\mathbf{w}_i^T\mathbf{x}_1 + w_{i0}) + (1-\lambda)(\mathbf{w}_i^T\mathbf{x}_2 + w_{i0})$
        $= g_i(\lambda\mathbf{x}_1 + (1-\lambda)\mathbf{x}_2)$
      - $\therefore \lambda\mathbf{x}_1 + (1-\lambda)\mathbf{x}_2 \in R_i$

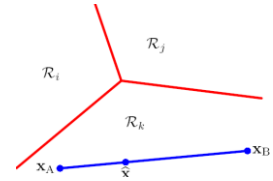


Fig. 4.3 [Bishop 06]

    - This restriction limits the flexibility and accuracy of the classifier
      - Suitable for cases with unimodel densities

Linear Discriminant Functions (7)

- Generalized linear discriminant functions
  - $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = w_l x_l + \cdots + w_1 x_1 + w_0; \quad \mathbf{w}, \mathbf{x} \in R^l$
    $= [\mathbf{w}^T \;\; w_0]\begin{bmatrix}\mathbf{x}\\ 1\end{bmatrix} = \mathbf{w}'^T\mathbf{x}'; \quad \mathbf{w}', \mathbf{x}' \in R^{l+1}$
  - The decision surfaces are l-dimensional hyperplanes passing through the origin of the $(l+1)$-dimensional expanded space
  - Example
    - $l = 2$
    - $g(\mathbf{x}) = \mathbf{w}'^T\mathbf{x}' = [w_1 \;\; w_2 \;\; w_0]\begin{bmatrix}x_1\\ x_2\\ 1\end{bmatrix}$

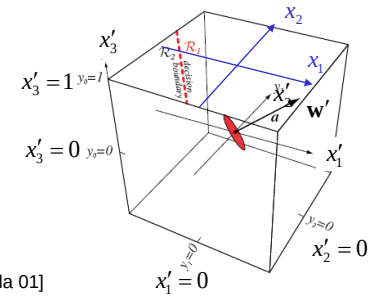


Fig. 5.7 [Duda 01]

Two-Category Linearly Separable Case (1)

- Linearly separable case
  - The samples can be correctly classified using linear classifiers
  - Given N labeled samples
  - To determine the weight $\mathbf{w}$ in $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$
    - Such that
      - $sign(\mathbf{w}^T\mathbf{x}_i) = +1 \quad \text{if} \quad \mathbf{x}_i \in C_1$
      - $sign(\mathbf{w}^T\mathbf{x}_i) = -1 \quad \text{if} \quad \mathbf{x}_i \in C_2$
- The steps
  - Define an appropriate cost function $J(\mathbf{w})$
  - Choose an algorithm to minimize the cost function
    - The solution may not be unique

Two-Category Linearly Separable Case (2)

- Solution vector (or separating vector) $\mathbf{w}$
  - Not unique
  - Additional constraint
    - To seek a unit length $\mathbf{w}$ to maximize the minimum distance from the samples to the separating plane, or
    - To seek the minimum-length $\mathbf{w}$ satisfying
      - $\mathbf{w}^T\mathbf{x}_i \geq z > 0, \ \forall \mathbf{x}_i \in C_1$
      - $\mathbf{w}^T\mathbf{x}_i \leq -z < 0, \ \forall \mathbf{x}_i \in C_2$

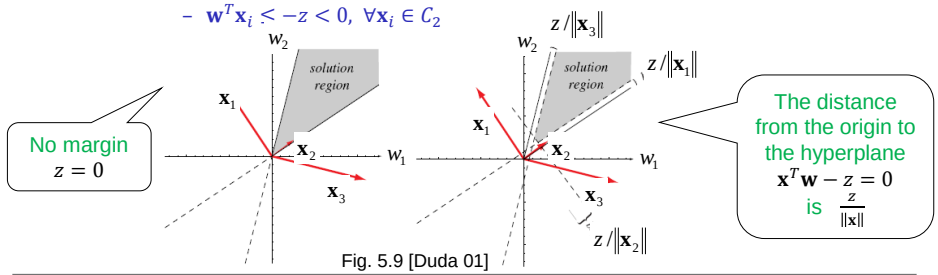


Fig. 5.9 [Duda 01]

Two-Category Linearly Separable Case (3)

- Gradient descent procedures
  - To minimize a criterion function $J(\mathbf{w})$
    - Initialize $\mathbf{w}(0)$
    - Move $\mathbf{w}(t+1)$ from $\mathbf{w}(t)$ in the direction of steepest descent (i.e. the negative of the gradient)
    - $\mathbf{w}(t+1) = \mathbf{w}(t) - \mu(t)\nabla J(\mathbf{w}(t))$
      - $\mu(t)$: the learning rate or step size

The gradient $\nabla J(\mathbf{w})$ points to the direction of maximum rate of increase of J at $\mathbf{w}$

  - Steepest descent method
    - The learning rate is chosen to achieve the max amount of decrease of the cost function at each individual step
      - $\mu(t) = \underset{\mu>0}{\operatorname{argmin}} J(\mathbf{w}(t) - \mu\nabla J(\mathbf{w}(t)))$
    - That is, from $\mathbf{w}(t)$, we conduct a line search in the direction $-\nabla J(\mathbf{w})$ until a minimizer $\mathbf{w}(t+1)$ is found

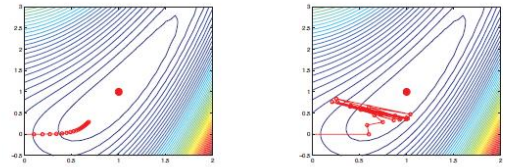
Two-Category Linearly Separable Case (4)

- Newton's descent
  - Given a starting point $\mathbf{w}(t)$
  - Construct a quadratic approximation to the criterion function that matches the 1st and 2nd-derivative at that point
    - $J(\mathbf{w}(t+1)) \approx J(\mathbf{w}(t)) + \nabla J^T(\mathbf{w}(t))(\mathbf{w}(t+1) - \mathbf{w}(t)) + \frac{1}{2}(\mathbf{w}(t+1) - \mathbf{w}(t))^T \mathbf{H}_{\mathbf{w}=\mathbf{w}(t)}(\mathbf{w}(t+1) - \mathbf{w}(t))$
  - Minimizing the approximate function to obtain $\mathbf{w}(t+1)$
    - Let $\frac{\partial J(\mathbf{w}(t+1))}{\partial \mathbf{w}(t+1)} = 0$
    - $\Rightarrow \nabla J(\mathbf{w}(t)) + \frac{1}{2}2\mathbf{H}(\mathbf{w}(t))(\mathbf{w}(t+1) - \mathbf{w}(t)) = 0$
    - $\Rightarrow \mathbf{w}(t+1) = \mathbf{w}(t) - \mathbf{H}^{-1}(\mathbf{w}(t))\nabla J(\mathbf{w}(t))$

$\mathbf{H} = \left[\frac{\partial^2 J}{\partial w_i \partial w_j}\right]$

The Hessian matrix (the 2nd derivative)

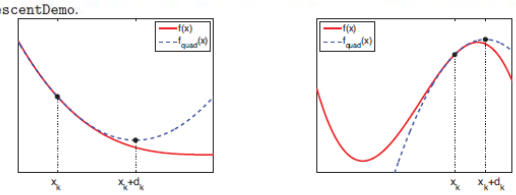
Fig. 8.2 [Murphy]



(a) (b)

Figure 8.2 Gradient descent on a simple function, starting from $(0, 0)$, for 20 steps, using a fixed learning rate (step size) η. The global minimum is at $(1, 1)$. (a) $\eta = 0.1$. (b) $\eta = 0.6$. Figure generated by `steepestDescentDemo`.

Fig. 8.4 [Murphy]



(a) (b)

Figure 8.4 Illustration of Newton's method for minimizing a 1d function. (a) The solid curve is the function $f(x)$. The dotted line $f_{quad}(x)$ is its second order approximation at x_k. The Newton step d_k is what must be added to x_k to get to the minimum of $f_{quad}(x)$. Based on Figure 13.4 of (Vandenberghe 2006). Figure generated by `newtonsMethodMinQuad`. (b) Illustration of Newton's method applied to a nonconvex function. We fit a quadratic around the current point x_k and move to its stationary point, $x_{k+1} = x_k + d_k$. Unfortunately, this is a local maximum, not minimum. This means we need to be careful about the extent of our quadratic approximation. Based on Figure 13.11 of (Vandenberghe 2006). Figure generated by `newtonsMethodNonConvex`.

Two-Category Linearly Separable Case (5)

Fig. 5.10 [Duda 01]

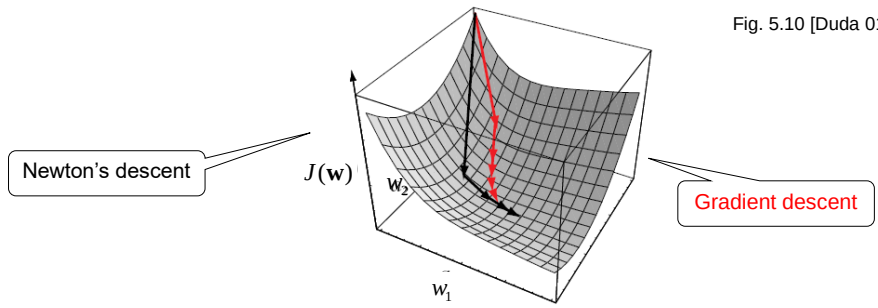


FIGURE 5.10. The sequence of weight vectors given by a simple gradient descent method (red) and by Newton's (second order) algorithm (black). Newton's method typically leads to greater improvement per step, even when using optimal learning rates for both methods. However the added computational burden of inverting the Hessian matrix used in Newton's method is not always justified, and simple gradient descent may suffice. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Two-Category Linearly Separable Case (6)

- Initialization $\mathbf{w}(0)$
  - Randomly initialize the weights
    - Choosing each weight independently from $N(0, small\ \sigma^2)$ usually works well
- Termination
  - Combination of the criteria
    - An upper bound for the number of iterations
    - A lower bound for the size of the gradient $\|J(\mathbf{w})\|$
    - A lower bound for the cost $J(\mathbf{w})$

The Perceptron Algorithm (1)

- Idea
  - To find a weight vector $\mathbf{w}$ that minimizes the number of misclassified samples
    - The cost function is discontinuous and is difficult to optimize
- The perceptron cost
  - $J(\mathbf{w}) = \sum_{\mathbf{x}\in Y}(\delta_{\mathbf{x}}\mathbf{w}^T\mathbf{x})$

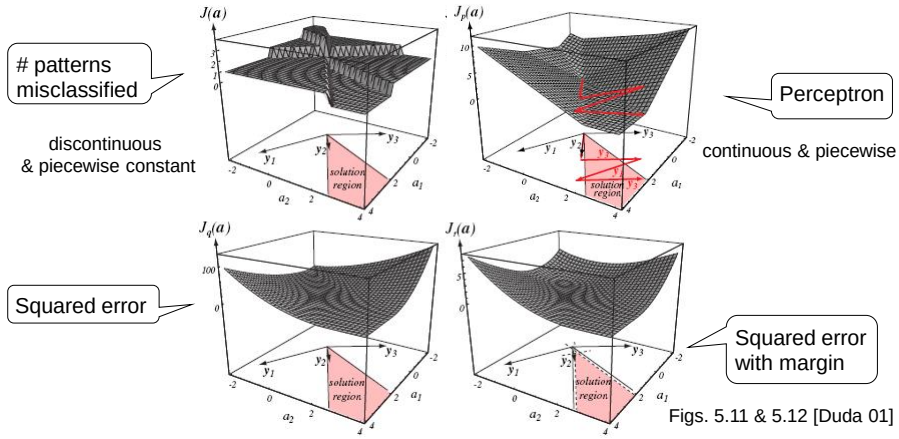
    Is proportion to the sum of the distances of the misclassified samples to the decision boundary
    - Y is the set of training vectors misclassified by $\mathbf{w}$
    - $\delta_{\mathbf{x}} = \begin{cases} -1, \mathbf{x} \in C_1 \\ +1, \mathbf{x} \in C_2 \end{cases}$
  - Thus
    - $J(\mathbf{w}) \geq 0$
    - $J(\mathbf{w}) = 0 \Leftrightarrow Y$ is empty

The Perceptron Algorithm (2)

- Minimizing the perceptron cost
  - Gradient descent method
    - The gradient vector
      - $\nabla J(\mathbf{w}) = \sum_{\mathbf{x}\in Y}(\delta_{\mathbf{x}}\mathbf{x})$
    - The updating
      - $\mathbf{w}(t+1) = \mathbf{w}(t) - \mu(t)\nabla J\big(\mathbf{w}(t)\big)$
        $= \mathbf{w}(t) - \mu(t)\sum_{\mathbf{x}\in Y}(\delta_{\mathbf{x}}\mathbf{x})$
      - The next weight vector is obtained by adding some multiple of the sum of the misclassified samples to the current weight vector

The Perceptron Algorithm (3)

- Example
  - 4 criteria as a function of weights in a linear classifier



Figs. 5.11 & 5.12 [Duda 01]

The Perceptron Algorithm (4)

- Example (p.95 [Theodoridis 09])
  - Assume at step t, there is only one misclassified sample $\mathbf{x}$
  - Let
    - $\mu(t) = 1$
    - $\mathbf{w}(t+1) = \mathbf{w}(t) - \sum_{\mathbf{x}\in Y_t}(\delta_{\mathbf{x}}\mathbf{x})$

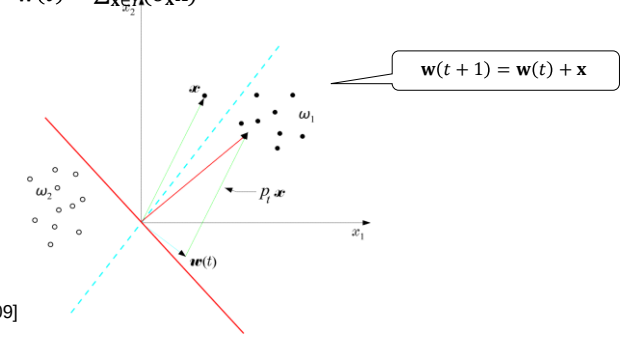


Fig. 3.2 [Theodoridis 09]

The Perceptron Algorithm (5)

- Example (p.97 [Theodoridis 09])
  - $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} = [w_1, w_2, w_0]\begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$
  - $\mathbf{w}(t+1) = \mathbf{w}(t) - \mu(t)\sum_{\mathbf{x}\in Y}(\delta_{\mathbf{x}}\mathbf{x}) = \mathbf{w}(t) - 0.7\sum_{\mathbf{x}\in Y}(\delta_{\mathbf{x}}\mathbf{x})$
  - If $\mathbf{w}(t) = [1, 1, -0.5]^T$
    - Two vectors are misclassified
      - $[0.4, 0.05]^T$, $[-0.2, 0.75]^T$
    - $\mathbf{w}(t+1) = \mathbf{w}(t) - 0.7\sum_{\mathbf{x}\in Y}(\delta_{\mathbf{x}}\mathbf{x})$

$$= \begin{bmatrix} 1 \\ 1 \\ -0.5 \end{bmatrix} - 0.7(-1)\begin{bmatrix} 0.4 \\ 0.05 \\ 1 \end{bmatrix} - 0.7(+1)\begin{bmatrix} -0.2 \\ 0.75 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1.42 \\ 0.51 \\ -0.5 \end{bmatrix}$$

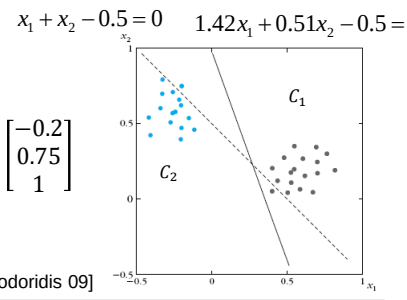


Fig. 3.3 [Theodoridis 09]

The Perceptron Algorithm (6)

- Proof of convergence
  - $\mathbf{w}(t+1) = \mathbf{w}(t) - \mu(t) \sum_{\mathbf{x} \in Y} (\delta_{\mathbf{x}} \mathbf{x})$
  - If training samples are linearly separable, the sequence of weight vectors will terminate at a solution vector
    - Let $\mathbf{w}^*$ be any solution vector,
      - i.e., $\mathbf{w}^{*T}\mathbf{x}_i > 0 \text{ if } \mathbf{x}_i \in C_1 \text{ and } \mathbf{w}^{*T}\mathbf{x}_i < 0 \text{ if } \mathbf{x}_i \in C_2$
    - Let
      - α be a positive scale factor
      - $\beta^2 = \max_{Y'} \left\| \sum_{\mathbf{x} \in Y'} (\delta_{\mathbf{x}} \mathbf{x}) \right\|^2$
      - $\gamma = \max_{Y'} \left(\sum_{\mathbf{x} \in Y'} (\delta_{\mathbf{x}} \mathbf{w}^{*T} \mathbf{x}) \right) < 0$
        - » The max value by considering all possible subsets of the available training feature vectors
      - $\mu(t) \geq 0$
        - » $\lim_{m \to \infty} \sum_{t=1}^{m} \mu(t) = \infty$ and $\lim_{m \to \infty} \sum_{t=1}^{m} \mu^2(t) < \infty$ $(e.g., \mu(t) = c/t)$

The Perceptron Algorithm (7)

- Proof of convergence (cont.)
  - Proof:
    - $\mathbf{w}(t+1) - \alpha \mathbf{w}^* = (\mathbf{w}(t) - \alpha \mathbf{w}^*) - \mu(t) \sum_{\mathbf{x} \in Y} (\delta_{\mathbf{x}} \mathbf{x})$
    - $\|\mathbf{w}(t+1) - \alpha \mathbf{w}^*\|^2$

$$= \|\mathbf{w}(t) - \alpha \mathbf{w}^*\|^2 - 2\mu(t)(\mathbf{w}(t) - \alpha \mathbf{w}^*)^T \sum_{\mathbf{x} \in Y} (\delta_{\mathbf{x}} \mathbf{x}) + \mu^2(t) \left\| \sum_{\mathbf{x} \in Y} (\delta_{\mathbf{x}} \mathbf{x}) \right\|^2$$

$$\leq \|\mathbf{w}(t) - \alpha \mathbf{w}^*\|^2 + 2\alpha\mu(t) \sum_{\mathbf{x} \in Y} (\delta_{\mathbf{x}} \mathbf{w}^{*T} \mathbf{x}) + \mu^2(t) \left\| \sum_{\mathbf{x} \in Y} (\delta_{\mathbf{x}} \mathbf{x}) \right\|^2$$

$$\leq \|\mathbf{w}(t) - \alpha \mathbf{w}^*\|^2 - 2\alpha\mu(t)|\gamma| + \mu^2(t)\beta^2$$

$\because -\sum_{\mathbf{x} \in Y} (\delta_{\mathbf{x}} \mathbf{w}^T(t) \mathbf{x}) < 0$

$$\beta^2 = \max_{Y'} \left\| \sum_{\mathbf{x} \in Y'} (\delta_{\mathbf{x}} \mathbf{x}) \right\|^2$$

$$\gamma = \max_{Y'} \left(\sum_{\mathbf{x} \in Y'} (\delta_{\mathbf{x}} \mathbf{w}^{*T} \mathbf{x}) \right) < 0$$

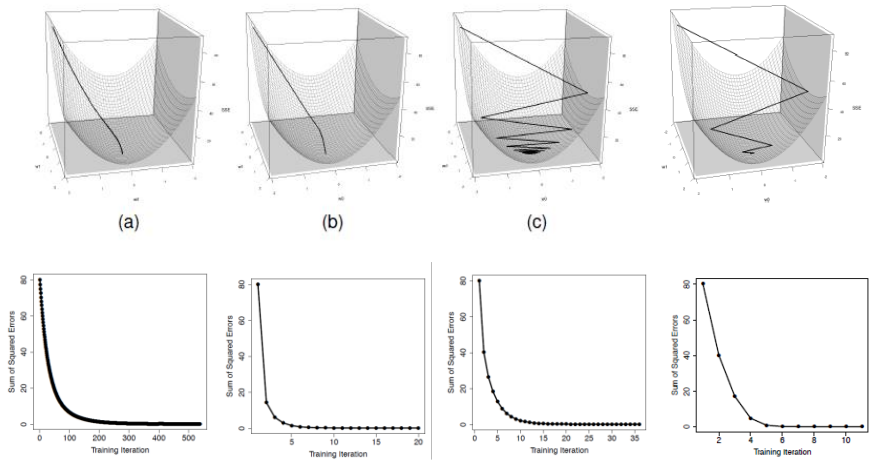
The Perceptron Algorithm (8)

- Proof of convergence (cont.)
  - If α is sufficiently large
    - Let $\alpha = \beta^2/2|\gamma|$
    - $\|\mathbf{w}(t+1) - \alpha\mathbf{w}^*\|^2 \le \|\mathbf{w}(t) - \alpha\mathbf{w}^*\|^2 - 2\alpha\mu(t)|\gamma| + \mu^2(t)\beta^2$
      $\le \|\mathbf{w}(t) - \alpha\mathbf{w}^*\|^2 + \beta^2(\mu^2(t) - \mu(t)|) \le \cdots$
      $\le \|\mathbf{w}(0) - \alpha\mathbf{w}^*\|^2 + \beta^2(\sum_{k=0}^{t}\mu^2(k) - \sum_{k=0}^{t}\mu(k))$
    - Then there will be a constant t_0 such that the right-hand side becomes non-positive
      - $0 \le \|\mathbf{w}(t_0+1) - \alpha\mathbf{w}^*\|^2 \le 0$
      - $\mathbf{w}(t_0+1) = \alpha\mathbf{w}^*$
    - Thus the sequence of corrections will terminate after a finite number of corrections

The Perceptron Algorithm (9)

- Variants of the perceptron algorithm
  - Single-sample correction
    - The N training vectors enter the algorithm cyclically
      - Updating the estimate as each new data point arrives
    - Variable increment
      - $\mathbf{w}(t+1) = \begin{cases} \mathbf{w}(t) + \mu(t)\mathbf{x}_t, & \text{if } \mathbf{x}_t \in C_1, \mathbf{w}^T(t)\mathbf{x}_t \le 0 \\ \mathbf{w}(t) - \mu(t)\mathbf{x}_t, & \text{if } \mathbf{x}_t \in C_2, \mathbf{w}^T(t)\mathbf{x}_t \ge 0 \\ \mathbf{w}(t), & \text{otherwise} \end{cases}$
    - Fixed increment
      - $\mu(t) = \mu = constant$
    - Variable-increment perceptron with margin
      - $\mathbf{w}(t+1) = \begin{cases} \mathbf{w}(t) + \mu(t)\mathbf{x}_t, & \text{if } \mathbf{x}_t \in C_1, \mathbf{w}^T(t)\mathbf{x}_t \le -z \\ \mathbf{w}(t) - \mu(t)\mathbf{x}_t, & \text{if } \mathbf{x}_t \in C_2, \mathbf{w}^T(t)\mathbf{x}_t \ge z \\ \mathbf{w}(t), & \text{otherwise} \end{cases}$

$\mu = 0.002$ $\mu = 0.08$ $\mu = 0.18$ $\mu(t) = 0.18 \times \frac{10}{10+t}$



[Kelleher et al., 2015]

Multi-Class Generalization (1)

- K-class problem
  - $g_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x};\ \ i = 1, \ldots, K;\ \mathbf{x} \in R^{l+1}$
    - $\mathbf{x} \rightarrow C_i$ if $\mathbf{w}_i^T\mathbf{x} > \mathbf{w}_j^T\mathbf{x},\ \forall j \neq i$
- Kesler's construction
  - Treating the $K-1$ inequalities, $\mathbf{w}_i^T\mathbf{x} - \mathbf{w}_j^T\mathbf{x} > 0, \forall j \neq i$ as
    - Determining the $(l+1)K$-dimensional weight vector $\mathbf{w}$ to correctly classify all $K-1$ vectors $\mathbf{x}_{ij}$
    - e.g., $\mathbf{x} \in C_i, i = 1 \Rightarrow \mathbf{w} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \mathbf{w}_3 \\ \vdots \\ \mathbf{w}_K \end{bmatrix}, \mathbf{x}_{i2} = \begin{bmatrix} \mathbf{x} \\ -\mathbf{x} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \mathbf{x}_{i3} = \begin{bmatrix} \mathbf{x} \\ \mathbf{0} \\ -\mathbf{x} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \ldots, \mathbf{x}_{iK} = \begin{bmatrix} \mathbf{x} \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ -\mathbf{x} \end{bmatrix}$
  - To design a linear classifier, in the $(l+1)K$ dimensional space
    - So that each of the $(K-1)N$ training vectors lies in its positive side

Multi-Class Generalization (2)

- Example 3.3 (p.102 [Theodoridis 09])
  - A 3-class problem in 2-D space
    - $C_1: \begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}2\\2\end{bmatrix}\begin{bmatrix}2\\1\end{bmatrix}$ $C_2: \begin{bmatrix}1\\-1\end{bmatrix}\begin{bmatrix}1\\-2\end{bmatrix}\begin{bmatrix}2\\-2\end{bmatrix}$ $C_3: \begin{bmatrix}-1\\1\end{bmatrix}\begin{bmatrix}-1\\2\end{bmatrix}\begin{bmatrix}-2\\1\end{bmatrix}$
  - To find $\mathbf{w}^T\mathbf{x} > 0$ in the 9-dimensional space $\mathbf{x} \in R^9$, $\mathbf{w}_i \in R^3$
    - $\mathbf{w} = \begin{bmatrix}\mathbf{w}_1\\\mathbf{w}_2\\\mathbf{w}_3\end{bmatrix}$; $\begin{bmatrix}1\\1\end{bmatrix} \Rightarrow \begin{bmatrix}1\\1\\1\\-1\\-1\\-1\\0\\0\\0\end{bmatrix}\begin{bmatrix}1\\1\\1\\0\\0\\0\\-1\\-1\\-1\end{bmatrix}$, $\begin{bmatrix}1\\-2\end{bmatrix} \Rightarrow \begin{bmatrix}-1\\2\\-1\\1\\-2\\1\\0\\0\\0\end{bmatrix}\begin{bmatrix}0\\0\\0\\1\\-2\\1\\-1\\2\\-1\end{bmatrix}$, $\begin{bmatrix}-2\\1\end{bmatrix} \Rightarrow \begin{bmatrix}2\\-1\\-1\\0\\0\\0\\-2\\1\\1\end{bmatrix}\begin{bmatrix}0\\0\\0\\2\\-1\\-1\\-2\\1\\1\end{bmatrix}$

Nonseparable Behavior

- Error-correction procedure
  - Modify $\mathbf{w}$ when and only when an error is encountered
    - However, the corrections can never cease in a nonseparable set
- Heuristic modification for nonseparable sets
  - To obtain suboptimal performance on nonseparable problems
  - By using a variable increment $\mu(t)$, with $\mu(t) \to 0$ as $t \to \infty$
    - e.g. $\mu(t)$ decrease as performance improves
    - e.g. $\mu(t) = \mu(1)/t$
  - By using Pocket algorithm to find the max. # of correct classification
    - Define a pocket weight $\mathbf{w}_s$ and initialize a counter h_s to zero
    - Let h = (# correctly classified training vectors using $\mathbf{w}(t+1)$)
      - Needs to evaluate all vectors using $\mathbf{w}(t+1)$
    - If $h > h_s$, then $\mathbf{w}_s \leftarrow \mathbf{w}(t+1)$, $h_s \leftarrow h$
    - Output $\mathbf{w}_s$

Least Squares Methods (1)

- Mean square error criterion function
  - Let the desired output be
    - $\mathbf{w}^T\mathbf{x} = y = \begin{cases} +1, & \text{if } \mathbf{x} \in C_1 \\ -1, & \text{if } \mathbf{x} \in C_2 \end{cases}$
  - The criterion
    - $J(\mathbf{w}) = E\{(y - \mathbf{w}^T\mathbf{x})^2\} = \int (y - \mathbf{w}^T\mathbf{x})^2 p(\mathbf{x})d\mathbf{x}$
      $= P(C_1)\int (1 - \mathbf{w}^T\mathbf{x})^2 p(\mathbf{x}|C_1)d\mathbf{x} + P(C_2)\int (1 + \mathbf{w}^T\mathbf{x})^2 p(\mathbf{x}|C_2)d\mathbf{x}$
    - Let $\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = -2E\{\mathbf{x}(y - \mathbf{w}^T\mathbf{x})\} = 0$
    - $\Rightarrow E\{\mathbf{x}y\} = E\{\mathbf{x}\mathbf{x}^T\}\hat{\mathbf{w}}$
    - $\Rightarrow \hat{\mathbf{w}} = R_{\mathbf{x}}^{-1}E\{\mathbf{x}y\} = E\{\mathbf{x}\mathbf{x}^T\}$

    Autocorrelation matrix $R_{\mathbf{x}} = E\{\mathbf{x}\mathbf{x}^T\}$
    Cross-correlation vector $E\{\mathbf{x}y\}$
  - But we do not know the underlying distributions for computing the correlation matrix and cross-correlation vector
    - Otherwise, use Bayesian classifier!!

Least Squares Methods (2)

- Stochastic gradient descent (Least mean squares (LMS))
  - Instead of considering the full batch gradient
  - Use the iterative scheme without having the statistical information
    - Pick a training data point uniformly at random ('stochastic')
    - Consider only the error on this data point
    - $\mathbf{w}(t+1) = \mathbf{w}(t) + \mu(t)\mathbf{x}_t(y_t - \mathbf{x}_t^T\mathbf{w}(t))$

RSS (residual sum of errors)

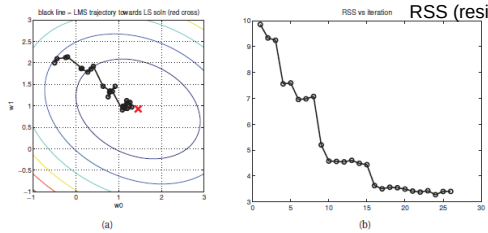


Figure 8.8 Illustration of the LMS algorithm. Left: we start from $\theta = (-0.5, 2)$ and slowly converging to the least squares solution of $\hat{\theta} = (1.45, 0.92)$ (red cross). Right: plot of objective function over time. Note that it does not decrease monotonically. Figure generated by LMSdemo.

Fig. 8.8 [Murphy]

Least Squares Methods (3)

- Sum-of-squared-error (SSE) criterion function
  - Let $\mathbf{w}^T\mathbf{x}_i = y_i = \begin{cases} +1, & \text{if } \mathbf{x}_i \in C_1 \\ -1, & \text{if } \mathbf{x}_i \in C_2 \end{cases}$
  - The criterion
    - Sum of the error squares over ALL available training samples
      - No need of explicit knowledge of the underlying pdfs
    - $J(\mathbf{w}) = \sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x}_i)^2 = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$
      - The data matrix $\boldsymbol{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix}_{N \times l}$
      - The error vector $\mathbf{e} = \mathbf{y} - \mathbf{X}\mathbf{w}$
    - $\frac{\partial J(\boldsymbol{w})}{\partial \boldsymbol{w}} = -\mathbf{X}^{\mathbf{T}}(\mathbf{y} - \mathbf{X}\mathbf{w}) = 0$
      - $\hat{\mathbf{w}} = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y} = \mathbf{X}^{+}\mathbf{y}$

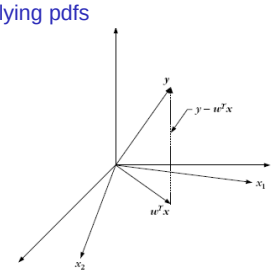


Fig. 3.6 [Theodoridis 09]

Least Squares Methods (4)

- Example 3.4 [Theodoridis 09]
  - C_1: $\begin{bmatrix}0.4\\0.5\end{bmatrix}\begin{bmatrix}0.6\\0.5\end{bmatrix}\begin{bmatrix}0.1\\0.4\end{bmatrix}\begin{bmatrix}0.2\\0.7\end{bmatrix}\begin{bmatrix}0.3\\0.3\end{bmatrix}$
  - C_2: $\begin{bmatrix}0.4\\0.6\end{bmatrix}\begin{bmatrix}0.6\\0.2\end{bmatrix}\begin{bmatrix}0.7\\0.4\end{bmatrix}\begin{bmatrix}0.8\\0.6\end{bmatrix}\begin{bmatrix}0.7\\0.5\end{bmatrix}$
  - $\hat{\mathbf{w}} = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y} = \mathbf{X}^{+}\mathbf{y}$
    $= \mathbf{X}^{+}[1\ 1\ 1\ 1\ 1\ -1\ -1\ -1\ -1\ -1]^{T}$
    $= \begin{bmatrix}-3.22\\0.24\\1.43\end{bmatrix}$

$-3.22x_1 + 0.24x_2 + 1.431 = 0$

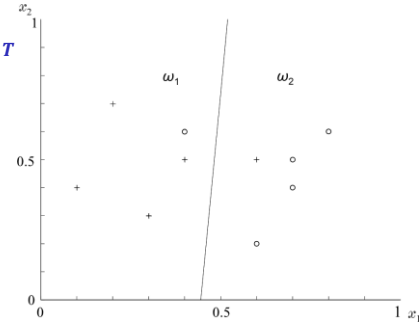
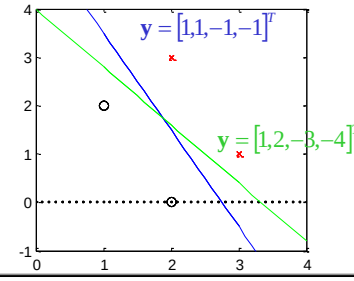


Fig. 3.7 [Theodoridis 09]

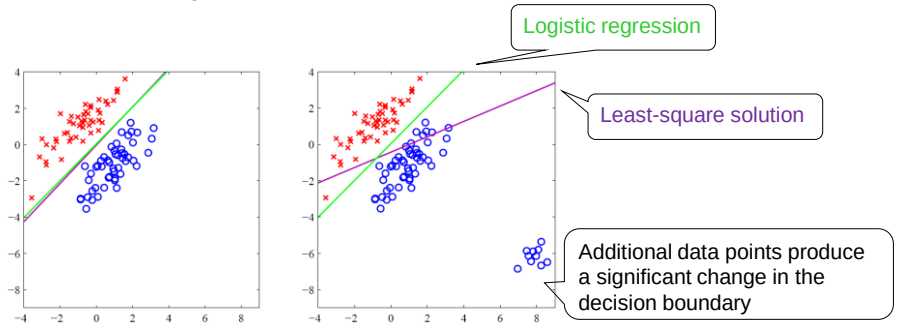
Least Squares Methods (5)

- Sum-of-squared-error criterion function (cont.)
  - Example (p. 241, [Duda 01])
    - Given 4 training points
      - $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T & 1 \\ \mathbf{x}_2^T & 1 \\ \mathbf{x}_3^T & 1 \\ \mathbf{x}_4^T & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 0 & 1 \\ 3 & 1 & 1 \\ 2 & 3 & 1 \end{bmatrix}$, $\mathbf{X}^+ = (\mathbf{X^T X})^{-1}\mathbf{X^T} = \begin{bmatrix} -1/2 & -1/6 & 1/2 & 1/6 \\ 0 & -1/3 & 0 & 1/3 \\ 5/4 & 13/12 & -4/3 & -7/12 \end{bmatrix}$
    - If $\mathbf{y} = [1, 1, -1, -1]^T$
      - $\hat{\mathbf{w}} = \mathbf{X}^+\mathbf{y} = \left[-\frac{4}{3}, -\frac{2}{3}, \frac{11}{3}\right]^T$
      - $g(\mathbf{x}) = -\frac{4}{3}x_1 - \frac{2}{3}x_2 + \frac{11}{3}$
    - If $\mathbf{y} = [1, 2, -3, -4]^T$
      - $\hat{\mathbf{w}} = \mathbf{X}^+\mathbf{y} = [-3, -2{,}8]^T$
      - $g(\mathbf{x}) = -3x_1 - 2x_2 + 8$



Least Squares Methods (6)

- Sum-of-squared-error criterion function (cont.)
  - Example (p. 186-187, [Bishop 06])
    - The least-square solutions lack robustness to outliers
      - Because all errors are squared and then summed up
      - Large error values influence the result much more than the small errors



Least Squares Methods (7)

- Multiclass generalization
  - To design K linear discriminant functions
    - $g_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x};\ \ i = 1, \dots, K;\ \mathbf{x} \in R^{l+1}$
    - The output response for each input vector is
      - $\mathbf{y} = [y_1, \dots, y_K]^T$
      - $y_j = \begin{cases} 1, & \text{if } \mathbf{x} \in C_j \\ 0, & \text{otherswise} \end{cases}$

$\mathbf{W}_{(l+1)\times K} = [\mathbf{w}_1, \dots, \mathbf{w}_K]$

  - The sum of error squares criterion
    - $J(\mathbf{W}) = \sum_{i=1}^{N}(\mathbf{y}_i - \mathbf{W}^T\mathbf{x}_i)^2 = \sum_{i=1}^{N}\sum_{j=1}^{K}\left(y_{ij} - \mathbf{w}_j{}^T\mathbf{x}_i\right)^2$
      $= \sum_{j=1}^{K}\sum_{i=1}^{N}\left(y_{ij} - \mathbf{w}_j{}^T\mathbf{x}_i\right)^2 = \sum_{j=1}^{K} J(\mathbf{w}_j)$
    - Equivalent to K independent minimization problems
      - Each one with desired output 1 for vectors belonging to the corresponding class and 0 for all the others

Logistic Discrimination (1)

- Logistic discrimination (or logistic regression)
  - Unlike the SSE method, which uses the hard threshold on $\mathbf{w}^T\mathbf{x}$ to produce +1 or -1 output
    - i.e., $y = \mathbf{w}^T\mathbf{x} = \begin{cases} +1, & \text{if } \mathbf{x} \in C_1 \\ -1, & \text{if } \mathbf{x} \in C_2 \end{cases}$
  - The output is mapped to a value between 0 and 1
    - Can be interpreted as a probability for the binary classification
      - $\mathbf{x} \in C_1 \Rightarrow \sigma(\mathbf{w}^T\mathbf{x}) > 0.5$
    - $\sigma(\cdot)$ is the logistic sigmoid function defined by
      - $\sigma(x) \equiv \frac{1}{1+\exp(-x)} = \frac{\exp(x)}{1+\exp(x)}$
        - » $\sigma(-\infty) = 0;\ \sigma(0) = 0.5; \sigma(\infty) = 1$
      - $1 - \sigma(x) = \sigma(-x) = \frac{1}{1+\exp(x)}$
      - $\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$

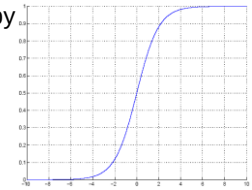


Fig 10.5 [Alpaydin, 2020]

Logistic Discrimination (2)

$\mathbf{x} \in C_1 \Rightarrow \sigma(\mathbf{w}^T\mathbf{x}) > 0.5$
$\mathbf{x} \in C_2 \Rightarrow \sigma(\mathbf{w}^T\mathbf{x}) < 0.5$

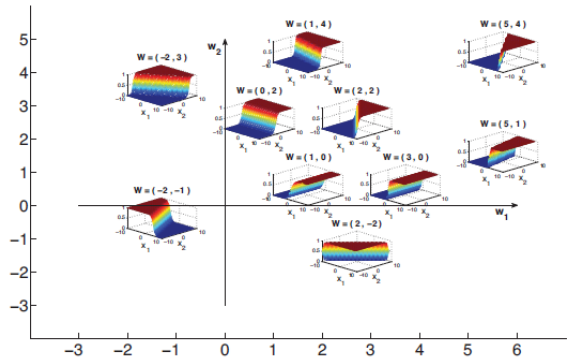


Fig. 8.1 [Murphy]

Figure 8.1 Plots of $\text{sigm}(w_1x_1 + w_2x_2)$. Here $\mathbf{w} = (w_1, w_2)$ defines the normal to the decision boundary. Points to the right of this have $\text{sigm}(\mathbf{w}^T\mathbf{x}) > 0.5$, and points to the left have $\text{sigm}(\mathbf{w}^T\mathbf{x}) < 0.5$. Based on Figure 39.3 of (MacKay 2003). Figure generated by `sigmoidplot2D`.

Logistic Discrimination (3)

- Logistic discrimination (cont.)
  - The logarithm of the likelihood ratio is modeled as linear
    - $\ln\frac{P(C_1|\mathbf{x})}{P(C_2|\mathbf{x})} = \mathbf{w}^T\mathbf{x}$ #parameters: $l+1$
      - We directly model the ratio but never explicitly estimate the class-conditional densities or prior
      - $\ln\frac{P(C_1|\mathbf{x})}{1-P(C_1|\mathbf{x})} = \mathbf{w}^T\mathbf{x} \Rightarrow P(C_1|\mathbf{x}) = \frac{\exp(\mathbf{w}^T\mathbf{x})}{1+\exp(\mathbf{w}^T\mathbf{x})}$
    - Thus, the posterior probabilities are
      - $P(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}); \quad P(C_2|\mathbf{x}) = 1 - P(C_1|\mathbf{x}) = \frac{1}{1+\exp(\mathbf{w}^T\mathbf{x})}$
  - This holds when the class-conditional densities are normal with a common covariance matrix (Ch3, case 3)
    - $P(C_1|\mathbf{x}) = \frac{p(\mathbf{x}|C_1)P(C_1)}{p(\mathbf{x}|C_1)P(C_1)+p(\mathbf{x}|C_2)P(C_2)} = \frac{\exp(\mathbf{w}^T\mathbf{x}+w_0)}{1+\exp(\mathbf{w}^T\mathbf{x}+w_0)}$
      - $\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$
      - $w_0 = \frac{1}{2}\left(\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2\right) + \ln\frac{P(C_1)}{P(C_2)}$
    - #parameters (mean+covariance+prior): $2l + \frac{l(l+1)}{2} + 1 = \frac{l(l+5)}{2} + 1$

Logistic Discrimination (4)

- Parameter estimation by ML
  - Given the training data set
    - $X = \{(\mathbf{x}_i, y_i); i = 1, \dots, N\}$
  - The unknown parameters $\boldsymbol{\theta} = \{\mathbf{w}\}$
  - The log-likelihood
    - $y_i \in \{0,1\}$
      - $L(\boldsymbol{\theta}) = ln\, p(X|\boldsymbol{\theta}) = \ln(\prod_{k=1}^{N} p(\mathbf{x}_k|\boldsymbol{\theta}))$
        $= \ln(\prod_{k=1}^{N} P(C_1|\mathbf{x}_k, \boldsymbol{\theta})^{y_k}(1 - P(C_1|\mathbf{x}_k, \boldsymbol{\theta}))^{1-y_k})$
        $= \sum_{k=1}^{N}\{y_k \ln P(C_1|\mathbf{x}_k, \boldsymbol{\theta}) + (1 - y_k)\ln(1 - P(C_1|\mathbf{x}_k, \boldsymbol{\theta}))\}$
    - Or $\tilde{y}_i \in \{+1, -1\}$
      - $L(\boldsymbol{\theta}) = \sum_{k=1}^{N} \ln \frac{1}{1+\exp(-\tilde{y}_i \mathbf{w}^T \mathbf{x}_k)}$

Logistic Discrimination (5)

- Parameter estimation by ML (cont.)
  - To maximize the log-likelihood => minimize the negative log-likelihood (NLL)
    - $y_i \in \{0,1\}$
      - $NLL(\boldsymbol{\theta}) = -L(\boldsymbol{\theta}) = -\sum_{k=1}^{N}\{y_k \ln \sigma(\mathbf{w}^T \mathbf{x}_k) + (1 - y_k)\ln(1 - \sigma(\mathbf{w}^T \mathbf{x}_k))\}$
    - $\tilde{y}_i \in \{+1, -1\}$
      - $NLL(\boldsymbol{\theta}) = -L(\boldsymbol{\theta}) = -\sum_{k=1}^{N} \ln \frac{1}{1+\exp(-\tilde{y}_i \mathbf{w}^T \mathbf{x}_k)} = \sum_{k=1}^{N} \ln\{1 + \exp(-\tilde{y}_i \mathbf{w}^T \mathbf{x}_k)\}$
  - Need to use an optimization procedure to perform the minimization
    - e.g., Gradient descent
      - $\nabla_{\mathbf{w}} NLL(\boldsymbol{\theta}) = \sum_{k=1}^{N}(\sigma(\mathbf{w}^T \mathbf{x}_k) - y_k)\mathbf{x}_k$
      - Or $\nabla_{\mathbf{w}} NLL(\boldsymbol{\theta}) = \sum_{k=1}^{N} \frac{-\tilde{y}_i \mathbf{x}_k}{1+\exp(-\tilde{y}_i \mathbf{w}^T \mathbf{x}_k)}$
      - $\mathbf{w}(t+1) = \mathbf{w}(t) - \mu(t)\nabla_{\mathbf{w}} NLL(\mathbf{w}(t))$

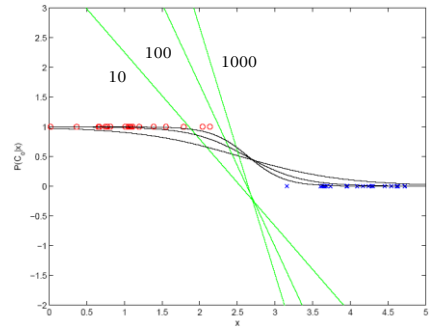
Logistic Discrimination (6)

For $j = 0, \ldots, d$
  $w_j \leftarrow$ rand(-0.01,0.01)
Repeat
  For $j = 0, \ldots, d$
    $\Delta w_j \leftarrow 0$
  For $t = 1, \ldots, N$
    $o \leftarrow 0$
    For $j = 0, \ldots, d$
      $o \leftarrow o + w_j x_j^t$
    $y \leftarrow \text{sigmoid}(o)$
    $\Delta w_j \leftarrow \Delta w_j + (r^t - y) x_j^t$
  For $j = 0, \ldots, d$
    $w_j \leftarrow w_j + \eta \Delta w_j$
Until convergence

$y = \sigma(\mathbf{w}^T \mathbf{x}_j)$
r: label indicator

Fig 10.6 [Alpaydin, 2020]

Line: $wx + w_0 = 0$ after 10, 100, 1000 iterations



$P(C_o|x)$ of the class o after 10, 100, 1000 iterations

Fig 10.7 [Alpaydin, 2020]

Support Vector Machines (1)

- Two-category separable case
  - The perceptron rule with margin seeks a solution that
    - $\mathbf{w}^T \mathbf{x}_i + w_0 \geq +z, \ \forall \mathbf{x}_i \in C_1$
    - $\mathbf{w}^T \mathbf{x}_i + w_0 \leq -z, \ \forall \mathbf{x}_i \in C_2$
  - If we scale z, w_0 and $\mathbf{w}$ so that
    - $\mathbf{w}^T \mathbf{x}_i + w_0 \geq +1, \ \forall \mathbf{x}_i \in C_1$
    - $\mathbf{w}^T \mathbf{x}_i + w_0 \leq -1, \ \forall \mathbf{x}_i \in C_2$
  - Then the distance between the separating hyperplane H and each of the two parallel hyperplanes (H_1 and H_2) is $\frac{1}{\|\mathbf{w}\|}$
    - $H: g(\mathbf{x}) = \mathbf{w}^T \mathbf{x}_i + w_0 = 0$
    - $H_1: \mathbf{w}^T \mathbf{x}_i + w_0 = +1$
    - $H_2: \mathbf{w}^T \mathbf{x}_i + w_0 = -1$

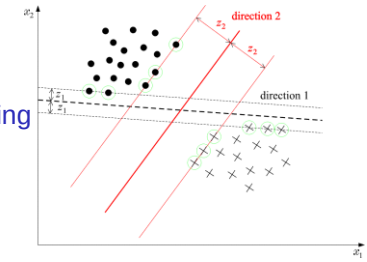


Fig. 3.10 [Theodoridis 09]

Support Vector Machines (2)

- Two-category separable case (cont.)
  - Margin
    - The smallest distance to two parallel hyperplanes on each side of the separating hyperplane

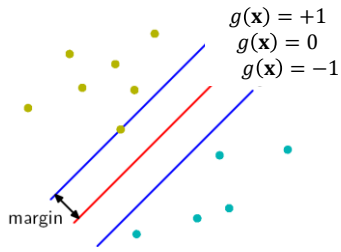
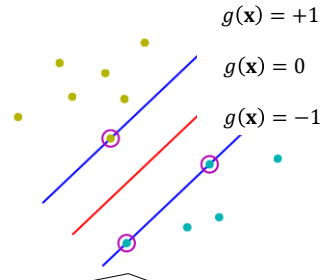


Fig. 7.1 [Bishop 06]



Location of the decision boundary is determined by a subset of the data points, known as support vectors

Support Vector Machines (3)

- Two-category separable case (cont.)
  - Maximal margin classifier
    - To search for the hyperplane that gives the maximum possible margin
      - Maximize the margin $\frac{1}{\|\mathbf{w}\|} + \frac{1}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$
      - Requiring that
        - » $\mathbf{w}^T\mathbf{x}_i + w_0 \geq +1, \ \forall \mathbf{x}_i \in C_1$
        - » $\mathbf{w}^T\mathbf{x}_i + w_0 \leq -1, \ \forall \mathbf{x}_i \in C_2$
    - Let y_i be the class indicator
    - Then the problem becomes
      - Minimize $J(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2}$
      - Subject to $y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq +1; \quad i = 1, \ldots, N$
        - » $y_i = \begin{cases} +1, & \text{if } \mathbf{x}_i \in C_1 \\ -1, & \text{if } \mathbf{x}_i \in C_2 \end{cases}$

Nonlinear (quadratic) optimization with linear inequality constraints

Support Vector Machines (4)

- Optimization for constrained problem (Appendix C.4 [Theodoridis 09])
  - The primal problem
    - Minimize $J(\boldsymbol{\theta})$
    - Subject to $f_i(\boldsymbol{\theta}) \geq 0; \quad i = 1, \ldots, N$
  - Lagrangian function
    - To augment the objective function with a weighted sum of the constraint functions
    - $L(\boldsymbol{\theta}, \boldsymbol{\lambda}) = J(\boldsymbol{\theta}) - \sum_{i=1}^{N} \lambda_i f_i(\boldsymbol{\theta})$
      - $\{\lambda_i; i = 1, \ldots, N\}$: the Lagrange multipliers
      - $\boldsymbol{\lambda}$: the dual variables or Lagrange multiplier vectors

Support Vector Machines (5)

- Optimization for constrained problem (cont.)
  - The dual function
    - $D(\boldsymbol{\lambda}) = \inf_{\boldsymbol{\theta}} L(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \inf_{\boldsymbol{\theta}} \left(J(\boldsymbol{\theta}) - \sum_{i=1}^{N} \lambda_i f_i(\boldsymbol{\theta}) \right)$
  - Suppose $\tilde{\boldsymbol{\theta}}$ is a feasible point of the primal problem and $\lambda_i \geq 0$
    - That is, $f_i(\tilde{\boldsymbol{\theta}}) \geq 0, \quad \forall i$
    - Then
      - $-\sum_{i=1}^{N} \lambda_i f_i(\tilde{\boldsymbol{\theta}}) \leq 0$
      - $L(\tilde{\boldsymbol{\theta}}, \boldsymbol{\lambda}) = J(\tilde{\boldsymbol{\theta}}) - \sum_{i=1}^{N} \lambda_i f_i(\tilde{\boldsymbol{\theta}}) \leq J(\tilde{\boldsymbol{\theta}})$
    - Thus
      - $D(\boldsymbol{\lambda}) = \inf_{\boldsymbol{\theta}} L(\boldsymbol{\theta}, \boldsymbol{\lambda}) \leq L(\tilde{\boldsymbol{\theta}}, \boldsymbol{\lambda}) \leq J(\tilde{\boldsymbol{\theta}})$
  - The dual function yields lower bounds on the optimal value of the primal problem
    - $D(\boldsymbol{\lambda}) \leq p^*$

Support Vector Machines (6)

- Optimization for constrained problem (cont.)
  - The Lagrange dual problem
    - Maximize $D(\boldsymbol{\lambda})$
    - Subject to $\boldsymbol{\lambda} \geq 0$
  - The Lagrange dual problem is a convex optimization problem, whether or not the primal problem is convex
    - The objective is concave
    - The constraint is convex

Support Vector Machines (7)

- Karush-Kuhn-Tucker (KKT) conditions (C.4 [Theodoridis 09])
  - Given the optimization problem with inequality constraint
    - Minimize $J(\boldsymbol{\theta})$
    - Subject to $f_i(\boldsymbol{\theta}) \geq 0; \quad i = 1, \ldots, N$
  - Define the Lagrangian function
    - $L(\boldsymbol{\theta}, \boldsymbol{\lambda}) = J(\boldsymbol{\theta}) - \sum_{i=1}^{N} \lambda_i f_i(\boldsymbol{\theta})$
  - The necessary conditions for the existence of a local minimizer $\boldsymbol{\theta}^*$
    - $\frac{\partial}{\partial \boldsymbol{\theta}} L(\boldsymbol{\theta}, \boldsymbol{\lambda}) = 0$
      - The minimum must be a stationary point
    - $\lambda_i \geq 0; \quad i = 1, \ldots, N$
      - A constraint is called inactive if the corresponding $\lambda_i = 0$
    - $\lambda_i f_i(\boldsymbol{\theta}^*) = 0; \quad i = 1, \ldots, N$
      - At least one of the terms (λ_i or $f_i(\boldsymbol{\theta}^*)$) must be 0

Support Vector Machines (8)

- Two-category separable case (cont.)
  - The Lagrangian function (the primal form)
    - $L_p(\mathbf{w}, w_0, \boldsymbol{\lambda}) = \frac{\|\mathbf{w}\|^2}{2} - \sum_{i=1}^{N} \lambda_i (y_i(\mathbf{w}^T\mathbf{x}_i + w_0) - 1)$

parameters = $l + 1$

  - In the primal form, the KKT conditions are
    - $\frac{\partial}{\partial \mathbf{w}} L_p(\mathbf{w}, w_0, \boldsymbol{\lambda}) = \mathbf{w} - \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i = 0$
      - $\Rightarrow \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$

A linear combination of feature vectors (i.e. support vectors) with $\lambda_i > 0$

    - $\frac{\partial}{\partial w_0} L_p(\mathbf{w}, w_0, \boldsymbol{\lambda}) = -\sum_{i=1}^{N} \lambda_i y_i = 0$
      - $\Rightarrow \sum_{i=1}^{N} \lambda_i y_i = 0$
    - $\lambda_i \geq 0; \quad i = 1, \ldots, N$
    - $\lambda_i (y_i(\mathbf{w}^T\mathbf{x}_i + w_0) - 1) = 0; \quad i = 1, \ldots, N$

The support vectors lie on either of the two hyperplanes H_1 or H_2

Active constraint
$\lambda_i > 0 \Rightarrow y_i(\mathbf{w}^T\mathbf{x}_i + w_0) - 1 = 0$

Inactive constraint
$\lambda_i = 0 \Rightarrow y_i(\mathbf{w}^T\mathbf{x}_i + w_0) - 1 \geq 0$

Support Vector Machines (9)

- Two-category separable case (cont.)
  - Substituting $\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$ and $\sum_{i=1}^{N} \lambda_i y_i = 0$ into
    - $L_p(\mathbf{w}, w_0, \boldsymbol{\lambda}) = \frac{\|\mathbf{w}\|^2}{2} - \sum_{i=1}^{N} \lambda_i (y_i(\mathbf{w}^T\mathbf{x}_i + w_0) - 1)$
  - We have the dual form (see p.943-944 [Theodoridis 09])
    - Maximize $L_d(\boldsymbol{\lambda}) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$
    - Subject to
      - $\sum_{i=1}^{N} \lambda_i y_i = 0$
      - $\lambda_i \geq 0; \quad i = 1, \ldots, N$

The dual form expresses the criterion as inner product of patterns => key concept for nonlinear SVM!

parameters = N

Support Vector Machines (10)

- Two-category separable case (cont.)
  - Use numerical quadratic programming to solve the dual form constrained optimization problem
  - Once λ_i are obtained
  - Use support vectors (patterns with $\lambda_i > 0$) to solve w_0 and $\mathbf{w}$
    - $\mathbf{w} = \sum_{i \in SV} \lambda_i y_i \mathbf{x}_i$
    - w_0 is computed from all $\lambda_i (y_i (\mathbf{w}^T \mathbf{x}_i + w_0) - 1) = 0$ with $\lambda_i > 0$
      - e.g. an average over all support vectors
        - » $\mathbf{w}^T \mathbf{x}_i + w_0 = y_i$
        - » $\mathbf{w}^T \sum_{i \in SV} \mathbf{x}_i + N_{SV} w_0 = \sum_{i \in SV} y_i$
        - » $\Rightarrow w_0 = \frac{1}{N_{SV}} \sum_{i \in SV} y_i - \frac{1}{N_{SV}} \mathbf{w}^T \sum_{i \in SV} \mathbf{x}_i$
  - The linear discriminant function becomes
    - $g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$ (inner product)
      $= \sum_{i \in SV} \lambda_i y_i \mathbf{x}_i^T \mathbf{x} - \frac{1}{N_{SV}} \sum_{i \in SV} \sum_{j \in SV} \lambda_i y_i \mathbf{x}_i^T \mathbf{x}_j + \frac{1}{N_{SV}} \sum_{i \in SV} y_i$

Support Vector Machines (11)

- Example 3.5 ([Theodoridis 09])
  - Given the training data
    - $C_1: \begin{bmatrix}1\\1\end{bmatrix} \begin{bmatrix}1\\-1\end{bmatrix}$ and $C_2: \begin{bmatrix}-1\\1\end{bmatrix} \begin{bmatrix}-1\\-1\end{bmatrix}$
  - The Lagrangian function (the primal form)
    - $L_p(\mathbf{w}, w_0, \boldsymbol{\lambda}) = \frac{w_1^2 + w_2^2}{2} - \lambda_1 (w_1 + w_2 + w_0 - 1) - \lambda_2 (w_1 - w_2 + w_0 - 1)$
      $-\lambda_3 (w_1 - w_2 - w_0 - 1) - \lambda_4 (w_1 + w_2 - w_0 - 1)$
  - KKT conditions
    - $\frac{\partial}{\partial \mathbf{w}} L_p = 0 \Rightarrow \mathbf{w} = \begin{bmatrix}\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 \\ \lambda_1 - \lambda_2 - \lambda_3 + \lambda_4\end{bmatrix}$
    - $\frac{\partial}{\partial w_0} L_p = 0 \Rightarrow \lambda_1 + \lambda_2 - \lambda_3 - \lambda_4 = 0$
    - $\lambda_1 (w_1 + w_2 + w_0 - 1) = 0, \lambda_2 (w_1 - w_2 + w_0 - 1) = 0$
    - $\lambda_3 (w_1 - w_2 - w_0 - 1) = 0, \lambda_4 (w_1 + w_2 - w_0 - 1) = 0$
    - $\lambda_1, \lambda_2, \lambda_3, \lambda_4 \geq 0$

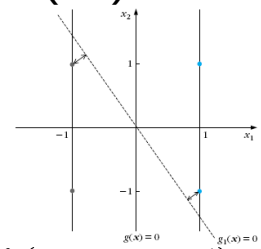


Fig. 3.12 [Theodoridis 09]

Support Vector Machines (12)

- Example 3.5 (cont.)
  - Assume the four constraints are active (i.e. $\lambda_1, \lambda_2, \lambda_3, \lambda_4 > 0$)
  - Then
    - $w_1 = \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4$
    - $w_2 = \lambda_1 - \lambda_2 - \lambda_3 + \lambda_4$
    - $w_1 + w_2 + w_0 - 1 = 0$
    - $w_1 - w_2 + w_0 - 1 = 0$
    - $w_1 - w_2 - w_0 - 1 = 0$
    - $w_1 + w_2 - w_0 - 1 = 0$
    - $\lambda_1 + \lambda_2 - \lambda_3 - \lambda_4 = 0$

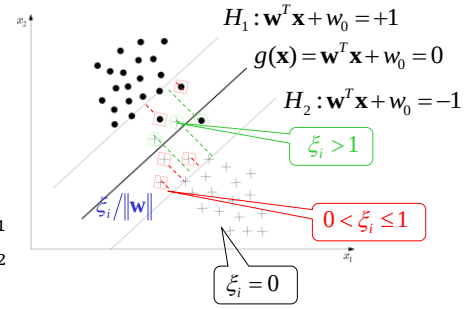
$$\Rightarrow \quad \begin{matrix} w_0 = 0 \\ \lambda_1 = 0.5 - \lambda_4 \\ \lambda_2 = \lambda_4 \\ \lambda_3 = 05 - \lambda_4 \end{matrix} \quad \Rightarrow \quad \begin{matrix} w_1 = 1 \\ w_2 = 0 \end{matrix}$$

  - Thus, the separating line is
    - $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = x_1 = 0$

Support Vector Machines (13)

- Linearly nonseparable case
  - Need to relax the constraints
    - $\mathbf{w}^T\mathbf{x}_i + w_0 \geq +1, \ \forall \mathbf{x}_i \in C_1$
    - $\mathbf{w}^T\mathbf{x}_i + w_0 \leq -1, \ \forall \mathbf{x}_i \in C_2$
  - To be
    - $\mathbf{w}^T\mathbf{x}_i + w_0 \geq +1 - \xi_i, \ \forall \mathbf{x}_i \in C_1$
    - $\mathbf{w}^T\mathbf{x}_i + w_0 \leq -1 + \xi_i, \ \forall \mathbf{x}_i \in C_2$
  - The slack variables $\xi_i \geq 0$
    - $\xi_i = 0$ for data points on or outside the correct margin boundary
    - $0 < \xi_i < 1$ for data points inside the correct margin
    - $\xi_i = 1$ for data points on the decision boundary
    - $\xi_i > 1$ for misclassified points

Fig. 3.11



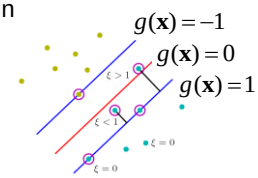


Fig. 7.3 [Bishop 06]

Support Vector Machines (14)

- Linearly nonseparable case (cont.)
  - The goal is to make the margin as large as possible but to keep the amount of misclassification as small as possible
    - Minimize $J(\mathbf{w}, \boldsymbol{\xi}) = \frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^{N}\xi_i$
    - Subject to
      - $y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq +1 - \xi_i; \quad i = 1, \ldots, N$
      - $\xi_i \geq 0; \quad i = 1, \ldots, N$

    Large C assigns a higher penalty to errors

  - The primal form of the Lagrangian
    - $L_p(\mathbf{w}, w_0, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\mu_i\xi_i - \sum_{i=1}^{N}\lambda_i(y_i(\mathbf{w}^T\mathbf{x}_i + w_0) - 1 + \xi_i)$
      - $\{\lambda_i \geq 0, \mu_i \geq 0; i = 1, \ldots, N\}$: the Lagrange multipliers

Support Vector Machines (15)

- Linearly nonseparable case (cont.)
  - The corresponding KKT conditions
    - $\frac{\partial}{\partial \mathbf{w}}L_p = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{N}\lambda_i y_i \mathbf{x}_i$
    - $\frac{\partial}{\partial w_0}L_p = 0 \Rightarrow \sum_{i=1}^{N}\lambda_i y_i = 0$
    - $\frac{\partial}{\partial \xi_i}L_p = 0 \Rightarrow C - \lambda_i - \mu_i = 0; \ \forall i$
    - $\lambda_i(y_i(\mathbf{w}^T\mathbf{x}_i + w_0) - 1 + \xi_i) = 0; \forall i$
    - $\mu_i\xi_i = 0 \Rightarrow (C - \lambda_i)\xi_i = 0; \forall i$
    - $\lambda_i \geq 0, \mu_i \geq 0; \forall i$

  Patterns with $0 < \lambda_i < C$ lie at the target distance of $\frac{1}{\|\mathbf{w}\|}$ from H and must have $\xi_i = 0$

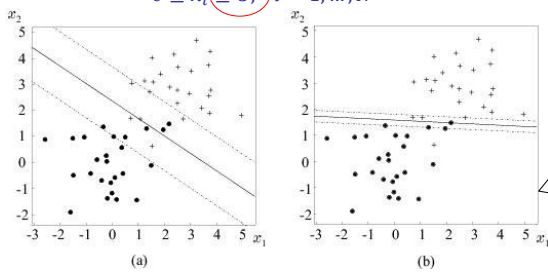
  Nonzero ξ_i can only occur when $\lambda_i = C$
  $\xi_i > 1 \Rightarrow \mathbf{x}_i$ is misclassified
  $0 < \xi_i < 1 \Rightarrow \mathbf{x}_i$ is classified correctly, but lies closer to H than $\frac{1}{\|\mathbf{w}\|}$

  - Thus
    - $0 \leq \lambda_i \leq C$
      - Support vectors $\lambda_i > 0$
        - » $0 < \lambda_i < C \Rightarrow \xi_i = 0$ (patterns on the margin)
        - » $\lambda_i = C$ (patterns on the margin or misclassified)

Support Vector Machines (16)

- Linearly nonseparable case (cont.)
  - The dual form becomes
    - Maximize $L_d(\boldsymbol{\lambda}) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$
    - Subject to
      - $\sum_{i=1}^{N} \lambda_i y_i = 0$
      - $0 \le \lambda_i \le C; \quad i = 1, \ldots, N$

The only difference with the separable case



(a) $C = 0.2$ (b) $C = 1000$

$\because J(\mathbf{w}, \boldsymbol{\xi}) = \frac{\|\mathbf{w}\|^2}{2} + C \sum_{i=1}^{N} \xi_i$

With larger C, the optimization process tries to reduce the number of points with $\xi_i > 0$ => Smaller margin

[Fig. 3.11, Theodoridis et. al.]